PERGAMON Computers and Mathematics with Applications 40 (2000) 1085–1096

An International Journal
computers & mathematics
with applications

www.elsevier.nl/locate/camwa

# The LFOPC Leap-Frog Algorithm for Constrained Optimization

J. A. SNYMAN
Multidisciplinary Design Optimization Group
Department of Mechanical and Aeronautical Engineering
University of Pretoria, Pretoria, 0002, South Africa
jan.snyman@eng.up.ac.za

*(Received and accepted May 1999)*

**Abstract**—This paper describes an accurate and reliable new algorithm (LFOPC) for solving constrained optimization problems, through a three-phase application of the well-established leap-frog method for unconstrained optimization, to penalty function formulations of the original constrained problems. The algorithm represents a considerable improvement over an earlier version (LFOPCON) which requires the judicious choice of parameter settings for efficient use. The current algorithm automatically executes normalization and scaling operations on the gradients of the constraints. This results in a robust algorithm that, apart from convergence tolerances, requires virtually no parameter settings. The method has been well tested, on both standard analytical test problems and practical engineering design problems. 

**Keywords**—Constrained optimization, Leap-frog method, Robust algorithm, Penalty function formulations.

## 1. INTRODUCTION

The dynamic leap-frog method for *unconstrained* optimization, originally proposed by Snyman [1,2], is a gradient method that generates a dynamic trajectory path, from any given starting point, towards a local optimum. This method differs conceptually from other gradient methods in that no explicit line searches are performed. The method uses only gradient information, with no explicit function evaluations being required. It is a proven robust method with the computed trajectory path being relatively insensitive to local inaccuracies and discontinuities in the gradients. This makes the method ideally suited for problems where the objective function contains experimental or numerical noise. The method, as embodied in the LFOP1(b) unconstrained optimization algorithm [2], has been widely applied to practical unconstrained optimization problems ranging from structures, mechanical systems, neural networks, chemistry, fluid flow, heat transfer, and composites. For a brief review of the different application areas of this *unconstrained* optimization algorithm up to 1993, the reader is referred to the paper of Snyman *et al.* [3]. The very recent paper of Holm and Botha [4] is also of particular interest because of the application of the method to the field of neural networks.

0898-1221/00/$ - see front matter  Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII: S0898-1221(00)00222-4

In papers [3,5], Snyman and his coworkers made a first attempt at adapting the method to solve *constrained* problems, by applying the LFOP1(b) algorithm for unconstrained optimization to penalty function formulations of the original constrained problems. In their adaptation, an overall penalty parameter is dynamically increased along the computed trajectory, starting with an initial value $\mu_0$ for the parameter, that is magnified by a factor $m$ at each step along the trajectory until a limit value $\mu_{\max}$ is attained. Thereafter, it is kept constant at this maximum value until convergence to an approximate solution is obtained.

The code LFOPCON [6], that implements the proposed adaptation, has successfully been applied to problems in optimal control [5] and structures [3], and very recently to the design of planar and three-dimensional manipulators [7–10] and mechanisms [11]. Currently, the code is also being used by the Automotive Research Center of the University of Michigan [12].

Unfortunately, the effective use of the LFOPCON code often requires experimentation to determine suitable values for the parameters $\mu_0$, $m$, and $\mu_{\max}$. If unsuitable choices are made for these parameters, the trajectory may either terminate *prematurely* at a suboptimal point or converge *very slowly* to a near optimum point. This paper reports on the development of a new and improved method for applying the leap-frog method to constrained problems. The resulting algorithm LFOPC (as opposed to the older version LFOPCON) requires almost no parameter settings (only two convergence tolerances and a maximum step size {optional} need be specified), and converges relatively quickly and reliably to a local minimum.

Here, test results are presented of the performance of LFOPC, when applied to a representative sample of standard and analytical test problems [13]. The results underline the reliability, accuracy, and robustness of LFOPC. Indeed, because of these characteristics, this latest method has *already* been used as the method of choice in solving optimization subproblems when approximation methods are applied to the optimization of complicated systems [14–17]. In some of these problems, because the function evaluations are the outcome of computationally very expensive analyses, a sequence of simpler subproblems is rather solved. The subproblems are constructed from the economic sampling of the behavior of the original and more complicated system. These subproblems are often inconsistent with no associated feasible regions. Even in these cases, the robust LFOPC algorithm succeeds in finding compromised solutions, from which the approximation process can further proceed.

Because of the already widespread application of the LFOPC algorithm to important constrained optimization engineering problems, and because of the increasing general interest in the leap-frog approach, it is necessary and timely that a detailed exposition be given of the background, theory, and construction of the new algorithm. This is the purpose of the current paper.

## 2. CONSTRAINED OPTIMIZATION PROBLEM

Consider the *constrained* optimization problem

$$\text{minimize } f(\mathbf{x}), \qquad \mathbf{x} \in \Re^n, \tag{1}$$

such that

$$g_i(\mathbf{x}) \leq 0, \qquad i = 1, 2, \ldots, m, \tag{2}$$

and

$$h_j(\mathbf{x}) = 0, \qquad j = 1, 2, \ldots, r, \tag{3}$$

where $f(\mathbf{x})$, $g_i(\mathbf{x})$, and $h_j(\mathbf{x})$ are scalar functions of $\mathbf{x}$, and are, respectively, referred to as the objective function, the inequality constraint, and equality constraint functions. The solution to this problem is denoted by $\mathbf{x}^*$.

The LFOPC algorithm, to be described in detail below, seeks the *constrained* optimum $\mathbf{x}^*$ of problem (1)–(3) by applying the LFOP1(b) (hereafter simply referred to as LFOP) leap-frog

dynamic *unconstrained* optimization algorithm of Snyman [1,2], to a penalty function formulation of the problem. This is done in three phases. In the first phase (Phase 0), using a moderate and standard penalty parameter value, the LFOP algorithm progresses relatively fast and smoothly to a point in the neighborhood of $\mathbf{x}^*$. In the next phase (Phase 1), with the penalty parameter significantly increased to a second standard value, the dynamic trajectory advances to a more accurate approximation to $\mathbf{x}^*$, and allows for the identification of the active set of constraints. Having identified the active set of constraints, the final phase (Phase 2) employs the LFOP algorithm to determine a least squares solution to the set of active constraints. In reaching this final solution, the trajectory effectively follows the shortest path from the previous approximation to $\mathbf{x}^*$, to the active constraints solution set. The point of convergence $\mathbf{x}^c$ is taken as the final approximation to $\mathbf{x}^*$.

## 3. THE LFOP ALGORITHM FOR UNCONSTRAINED MINIMIZATION

The detailed dynamic trajectory *leap-frog* algorithm for *unconstrained* minimization, used here for solving the penalty function formulated constrained problem associated with each of the three phases (0 to 2), remains essentially the same as that originally presented by Snyman [1,2].

The leap-frog method seeks the minimum of a function of $n$ variables by considering the associated dynamic problem of the motion of a particle of unit mass in a $n$-dimensional conservative force field, in which the potential energy of the particle at point $\mathbf{x}(t)$ at time $t$ is taken to be the function $f(\mathbf{x})$ to be minimized. In particular, the method requires the solution of the equations of motion of the particle subject to the specification of its initial position and velocity. The method computes an approximation to the associated trajectory by using the so-called *leap-frog* (Euler forward-Euler backward) method. An interfering strategy is applied that reduces the kinetic energy whenever the particle appears to move uphill. The consequence of this strategy is that a systematic reduction in the potential energy $f(\mathbf{x})$ is obtained, and the particle is forced to follow a path to a local minimum $\mathbf{x}^*$.

For further details and a fuller understanding of the trajectory method for unconstrained minimization, the reader is referred to the original papers [1,2]. The minor modifications applied here are as follows.

(i) The trajectory now also terminates if the step size becomes smaller than some prescribed positive tolerance $\varepsilon_x$, i.e., termination if $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon_x$.

(ii) The initial time step $\Delta t$ at starting point $\mathbf{x}_0$ is now no longer arbitrarily prescribed, but is computed from the initial gradient $\nabla f(\mathbf{x}_0)$ and the maximum prescribed step size $\delta$, according to the formula

$$\Delta t = \sqrt{\frac{\delta}{5\|\nabla f(\mathbf{x}_0)\|}}. \tag{4}$$

This guarantees an initial step size $\|\mathbf{x}_1 - \mathbf{x}_0\| = \delta/10$, where it is of course implicitly assumed that $\|\nabla f(\mathbf{x}_0)\| \neq 0$. If the latter is not true, $\mathbf{x}^* := \mathbf{x}_0$.

(iii) To prevent the occurrence of backtracking oscillations of the trajectory at maximum step size $\delta$, the time step is halved if the maximum step size is taken for more than five consecutive steps and with the direction of the gradient switching (i.e., $\mathbf{a}_{k+1}^{\mathrm{T}}\mathbf{a}_k < 0$) over the last step.

A detailed flowchart giving the essentials of the modified LFOP algorithm is shown in Figure 1. It is important to note that the only parameter settings required are the maximum step size $\delta$ and the convergence tolerances $\varepsilon_g$ and $\varepsilon_x$. Typically, $\delta$ is chosen to be of the order of the *diameter* of the region of interest, i.e., set $\delta := \sqrt{n}\, R_{\max}$, where $R_{\max}$ is the maximum variable range. If no choice is made, the default option $\delta = 1$ is used.

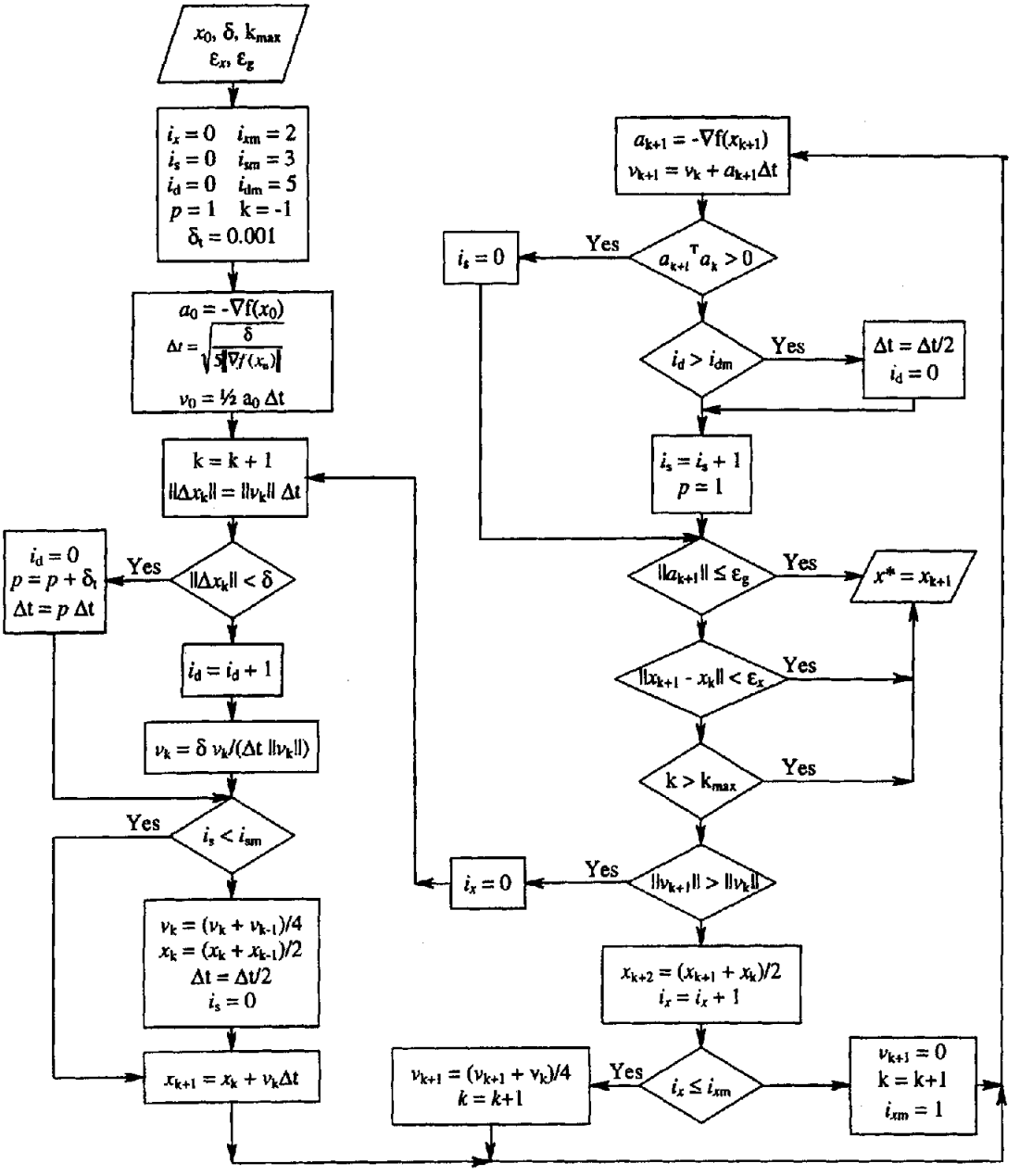



Figure 1. Flowchart of the LFOP unconstrained minimization algorithm.

## 4. PENALTY FUNCTION FORMULATION FOR THE LEAP-FROG METHOD

The solution to the constrained optimization problem (1)–(3) may be approximated by applying the unconstrained minimization algorithm LFOP to a penalty function formulation of the original problem. The appropriate penalty function to be minimized is

$$p(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \gamma f(\mathbf{x}) + \sum_{i=1}^{m} \alpha_i g_i^2(\mathbf{x}) + \sum_{j=1}^{r} \beta_j h_j^2(\mathbf{x}), \tag{5}$$

where $\gamma \equiv 1$ (unless otherwise stated) and the components of the vectors $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_m]^\top$ and $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_r]^\top$ are suitably chosen positive penalty parameters

$$\alpha_i = \begin{cases} \rho_i = \text{ a large positive number}, & \text{if } g_i(\mathbf{x}) > 0, \\ 0, & \text{if } g_i(\mathbf{x}) \leq 0, \end{cases}$$

and

$$\beta_j = \text{ a large positive number}. \tag{6}$$

Usually, the $\rho_i$ and $\beta_j$ take on the same positive value $\mu$, i.e., $\rho_i = \beta_j = \mu \gg 0$, for all values of $i$ and $j$.

Denote the penalty function and minimizer corresponding to the penalty parameter choice $\mu$ by $p(\mathbf{x}, \mu)$ and $\mathbf{x}^*(\mu)$, respectively. It can be shown [18] that, under relative general conditions, as $\mu$ tends to infinity the unconstrained minimum of $p(\mathbf{x}, \mu)$ tends to the constrained minimum of problem (1)–(3), i.e., the $\text{limit}_{\mu \to \infty}$ of $\mathbf{x}^*(\mu) = \mathbf{x}^*$.

Since LFOP is a gradient-only method, the algorithm needs only $\nabla p(\mathbf{x})$ given by

$$\nabla p(\mathbf{x}) = \gamma \nabla f(\mathbf{x}) + \sum_{i=1}^{m} 2\alpha_i g_i(\mathbf{x}) \nabla g_i(\mathbf{x}) + \sum_{j=1}^{r} 2\beta_j h_j(\mathbf{x}) \nabla h_j(\mathbf{x}). \tag{7}$$

Here, it is proposed to *normalize* each constraint term in (7) by dividing it, at each step, by the square of the norm of the associated constraint gradient vector (if the norm is nonzero). The gradient contributions are further *scaled* at each step by multiplying each constraint term by the norm of the gradient of the objective function term, i.e., by $\|\gamma \nabla f(\mathbf{x})\|$ if $\gamma \equiv 1$. These adjustments have the result that the effect of the spatial violation of a constraint on the penalty function gradient is approximately the same for all constraints. This avoids the necessity of selecting appropriately different penalty parameters for different constraints. The normalization and scaling also imply that, for a specific prescribed accuracy, the same overall penalty parameter $\mu$ may be used for different constrained problems. This eliminates the need for experimentation to determine parameter values sufficient to meet the prescribed accuracy of different problems. In order to avoid complications when the constrained minimum occurs at a point where the gradient of the objective function is close to zero, the norm of the gradient vector is increased by unity in setting the scaling factor; i.e., $\|\gamma \nabla f(\mathbf{x})\| + 1$ is used. The expression for the gradient of the penalty function used in practice is therefore given by

$$\nabla p(\mathbf{x}) = \gamma \nabla f(\mathbf{x}) + \sum_{i=1}^{m} 2\alpha_i'(\mathbf{x}) g_i(\mathbf{x}) \nabla g_i(\mathbf{x}) + \sum_{j=1}^{r} 2\beta_j'(\mathbf{x}) h_j(\mathbf{x}) \nabla h_j(\mathbf{x}), \tag{8}$$

where

$$\alpha_i'(\mathbf{x}) = \begin{cases} \dfrac{\mu(\|\gamma \nabla f(\mathbf{x})\| + 1)}{\|\nabla g_i(\mathbf{x})\|^2}, & \text{if } g_i(\mathbf{x}) > 0, \\ 0, & \text{if } g_i(\mathbf{x}) \leq 0, \end{cases} \tag{9}$$

and

$$\beta_j'(\mathbf{x}) = \frac{\mu(\|\gamma \nabla f(\mathbf{x})\| + 1)}{\|\nabla h_j(\mathbf{x})\|^2}. \tag{10}$$

The single parameter $\mu$ is chosen large enough to ensure sufficient accuracy. Its choice will be discussed later.

Note that since the norms of the gradient vectors of the objective and constraint functions are expected to be approximately constant in the neighborhood of the solution, the expression for the gradient of the penalty function (8) above effectively corresponds to a penalty function given by

$$p(\mathbf{x}) = \gamma f(\mathbf{x}) + \sum_{i=1}^{m} \alpha_i' g_i^2(\mathbf{x}) + \sum_{j=1}^{r} \beta_j' h_j^2(\mathbf{x}). \tag{11}$$

In order to avoid the penalty function from assuming excessive gradient values, which would have a negative effect on the functioning of the trajectory method, one further adjustment is required. If, for the normal choice $\gamma = 1$, the norm of the penalty function gradient (8) exceeds the norm of the objective function gradient plus one, then the penalty function gradient vector (8) is scaled so that its norm equals $\|\gamma\nabla f(\mathbf{x})\| + 1$; i.e., if $\|\nabla p(\mathbf{x})\| > \|\gamma\nabla f(\mathbf{x})\| + 1$, then set

$$\nabla p(\mathbf{x}) := \frac{(\gamma\|\nabla f(\mathbf{x})\| + 1)\nabla p(\mathbf{x})}{\|\nabla p(\mathbf{x})\|}. \tag{12}$$

## 5. THE LFOPC CONSTRAINED MINIMIZATION ALGORITHM

The Leap-Frog OPtimizer for Constrained problems, LFOPC, applies the LFOP unconstrained algorithm to penalty function formulations of the constrained problem (1)–(3) by using gradient expression (8) in three phases (Phases 0, 1, and 2).

### Phase 0

Given an initial starting point $\mathbf{x}_0$, then, with the overall penalty parameter set to $\mu := \mu_0$, apply LFOP using penalty function gradient (8) with $\gamma \equiv 1$. On convergence, this gives approximate optimum point $\mathbf{x}^*(\mu_0)$. Also, output on convergence is a final trajectory time step $\Delta t_f$. Identify the active inequality constraints at $\mathbf{x}^*(\mu_0)$, i.e., those that correspond to indices $i \in I_a$ where $I_a = \{i : i \in (1, 2, \ldots, m) \text{ and } g_i(\mathbf{x}^*(\mu_0)) > 0\}$.

Let $n_a$ denote the number of elements of $I_a$. If $m = 0$ and $r = 0$, then stop since no constraints specified and unconstrained optimum found. Also, if $n_a + r = 0$, then stop since no constraints active at $\mathbf{x}^*(\mu_0)$. Otherwise, if $n_a + r = n$ or $\|\nabla(\mathbf{x}^*(\mu_0))\| < 10^2\varepsilon_x$, go to Phase 2, else do Phase 1.

### Phase 1

With $\mathbf{x}_0 := \mathbf{x}^*(\mu_0)$, apply LFOP using penalty function gradient (8) with $\gamma \equiv 1$ and penalty parameter set to $\mu := \mu_1 \gg \mu_0$. Also, set initial time step $\Delta t := \Delta t_f$, output from Phase 0, instead of computing initial $\Delta t$ by formula (4). On satisfactory convergence, since $\mu_1 \gg \mu_0$, this gives a more accurate approximate solution $\mathbf{x}^*(\mu_1)$. Also, output is a corresponding new final time step $\Delta t_f$. Again, identify the indices of the active inequality constraints at $\mathbf{x}^*(\mu_1)$. This set $I_a$ may not necessarily be identical to the set identified in Phase 0. On convergence to $\mathbf{x}^*(\mu_1)$, go to Phase 2.

### Phase 2

With $\mathbf{x}_0 := \mathbf{x}^*(\mu)$, $\mu = \mu_0$, or $\mu = \mu_1$, depending on whether entry is from Phase 0 or Phase 1, and with initial time step $\Delta t = \Delta t_f$, apply LFOP using penalty function gradient (8) with $\gamma \equiv 0$, and

$$\alpha_i'(\mathbf{x}) = \begin{cases} \dfrac{\mu}{\|\nabla g_i(\mathbf{x})\|}, & \forall \mathbf{x} \text{ and } i \in I_a, \\ 0, & \forall \mathbf{x} \text{ and } i \notin I_a, \end{cases} \tag{13}$$

and

$$\beta_j'(\mathbf{x}) = \frac{\mu}{\|\nabla h_j(\mathbf{x})\|}, \qquad \forall \mathbf{x} \text{ and } \forall j. \tag{14}$$

This means that LFOP seeks a least-squares solution to the system of equations

$$\begin{aligned} g_i(\mathbf{x}) &= 0, \qquad I \in I_a, \\ h_j(\mathbf{x}) &= 0, \qquad j = 1, 2, \ldots, r. \end{aligned} \tag{15}$$

Since $\mathbf{x}^*(\mu)$, for sufficiently large $\mu$, should be very close to such a solution, and since $\Delta t$ is expected to be relatively small (output from converged Phase 0 or Phase 1), the trajectory

search of LFOP is in the neighborhood of $\mathbf{x}^*(\mu)$. This search is effectively in the hyperplane through $\mathbf{x}^*(\mu)$ spanned by the gradient vectors of all the active constraints at $\mathbf{x}^*(\mu)$ (see gradient expression (8) and remember $\gamma \equiv 0$). Thus, the point of convergence $\mathbf{x}^c$ is expected to be very close to a Karush-Kuhn-Tucker point, and this point is accordingly taken as the constrained optimum $\mathbf{x}^*$.

## 6. COMMENTS ON THE IMPLEMENTATION OF LFOPC

(i) Standard choice for penalty parameters: experimentation has shown that the choice of $\mu_0 = 10^2$ and $\mu_1 = 10^4$ guarantees satisfactory convergence, with sufficient accuracy, for a wide range of problems.

(ii) Choice of convergence tolerances: the choice $\varepsilon_g = 10^{-5}$ and $\varepsilon_x = 10^{-8}$ usually provides sufficient ($\sim$ eight significant digits) accuracy (see later section on results for test problems).

(iii) To improve speed of convergence and to avoid experimentation to determine suitable convergence tolerances, it is advisable to *normalize* $f(\mathbf{x})$ before implementation if $\|\nabla f(\mathbf{x}_0)\| \ll 1$. The suggested transformation is to set $f(\mathbf{x}) := f(\mathbf{x})/\|\nabla f(\mathbf{x}_0)\|$.

(iv) In some cases, when $n_a + r = n$ after Phase 0, it may not be advisable to perform the proposed bypass of Phase 1. The bypass is motivated by economic considerations, and is justified for cases where one expects that if $n$ active constraints are identified at $\mathbf{x}^*(\mu_0)$, that one may solve for a unique point of intersection to give $\mathbf{x}^*$. It is, however, possible that situations may arise where the $n$ identified active constraints may be inconsistent. It is also possible that if many constraints are almost active at $\mathbf{x}^*$, that the wrong set of $n$ active constraints may be selected (see the Appendix). If either of the above situations arise, it should be evident from the result of Phase 2, which will yield a slightly infeasible solution where a feasible $\mathbf{x}^*$ is expected. For all the test problems considered here, the suggested option of bypassing Phase 1 if $n_a + r = n$ was used without any complications.

(v) Care should be taken if after Phase 2 a slightly *compromised* solution is obtained. In such an event, restart with increased values for $\mu_0$ and $\mu_1$ (suggested values $\mu_0 = 10^3$ and $\mu_1 = 10^5$). The compromised situation arises if at $\mathbf{x}^*$ some inequality constraints are nearly but not exactly active. In particular, be careful if $n_a + r > n$. It may of course be that, because the problem is inconsistent, no feasible region exists, and that the only possibility is a compromised solution.

(vi) Finally, the option of bypassing Phase 1 if $\|\nabla f(\mathbf{x}^*(\mu_0))\| < 10^2 \varepsilon_g$ is justified, since, in this case, the relative flat objective function gradient at $\mathbf{x}^*(\mu_0)$ suggests that sufficient accuracy has been obtained to proceed to Phase 2.

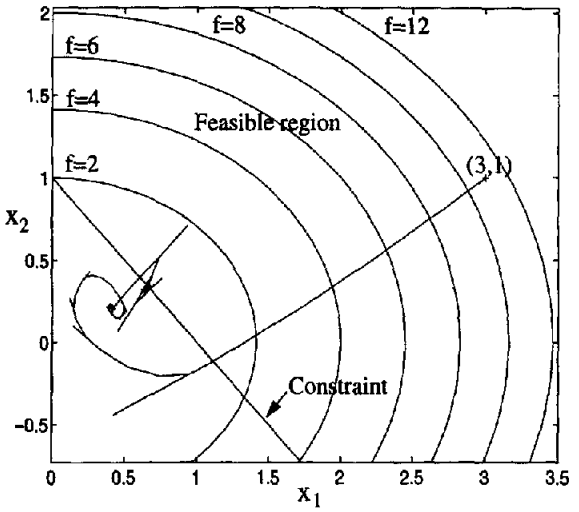
## 7. ILLUSTRATIVE EXAMPLE

The LFOPC algorithm is illustrated by its application to the following simple problem:

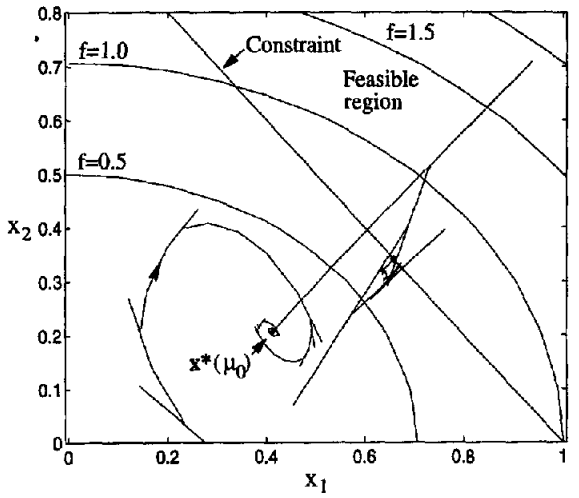
$$\begin{aligned} &\text{minimize } f(\mathbf{x}) = x_1^2 + 2x_2^2, \qquad \text{such that} \\ &\qquad g(\mathbf{x}) = -x_1 - x_2 + 1 \leq 0, \end{aligned} \tag{16}$$

with $\mathbf{x}_0 = [3, 1]^\top$. To accentuate the difference between $\mathbf{x}^*(\mu_0)$, $\mathbf{x}^*(\mu_1)$, and $\mathbf{x}^*$, relatively small values of $\mu_0 = 1$ and $\mu_1 = 10$ are used, instead of the standard choice: $\mu_0 = 10^2$ and $\mu_1 = 10^4$ recommended for higher accuracy. The progress of the algorithm is depicted in Figure 2a. Magnified views of the final part of the trajectory are presented in Figures 2b and 2c.

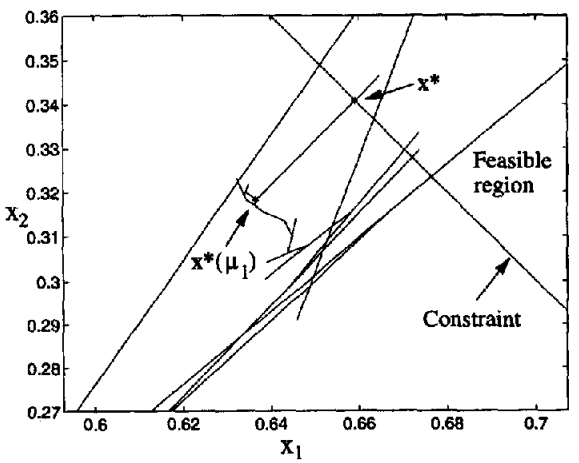
During Phase 0, as is apparent from the figures, the trajectory, starting at the point $\mathbf{x}_0 = [3, 1]^\top$, spirals towards and converges to the point $\mathbf{x}^*(\mu_0) = [0.413; 0.217]^\top$ with $f(\mathbf{x}^*(\mu_0)) = 0.255$. Characteristic of the trajectory path are the overshoot points where uphill motion occurs,




(a)




(b)




(c)

Figure 2. The (a) complete LFOPC trajectory for problem (16) with $\mathbf{x}_0 = [3, 1]^T$, and magnified views of the final part of the trajectory shown in (b) and (c).

Table 1. Performance of LFOPC and FMIN algorithms on standard test problems.

| Prob. No. [13] | $n$ | LFOPC | | FMIN [13] | |
|---|---|---|---|---|---|
| | | No. Steps = NPGE | Relative Error | NFE | Relative Error |
| 1 | 2 | 238 | $< 10^{-8}$ | 549 | $< 10^{-8}$ |
| 2 | 2 | 209* | $< 10^{-8}$ | 382 | $1.10^{-8}$ |
| 10 | 2 | 515 | $2.10^{-8}$ | 289 | $7.10^{-8}$ |
| 14 | 2 | 110 | $< 10^{-8}$ | 232 | $2.10^{-7}$ |
| 15 | 2 | 413* | $< 10^{-8}$ | 729 | $4.10^{-7}$ |
| 16 | 2 | 249 | $5.10^{-8}$ | 362 | $1.10^{-8}$ |
| 17 | 2 | 141 | $2.10^{-8}$ | 541 | $1.10^{-8}$ |
| 20 | 2 | 111 | $2.10^{-8}$ | 701* | $4.10^{-6}$ |
| 22 | 2 | 111 | $< 10^{-8}$ | 174 | $1.10^{-7}$ |
| 23 | 2 | 138 | $< 10^{-8}$ | 423 | $6.10^{-6}$ |
| 24 | 2 | 133 | $< 10^{-8}$ | 280 | $2.10^{-8}$ |
| 26 | 3 | 414 | $1.10^{-8}$ | 182 | $< 10^{-8}$ |
| 27 | 3 | 546 | $< 10^{-8}$ | 173 | $1.10^{-8}$ |
| 28 | 3 | 297 | $< 10^{-8}$ | 23 | $< 10^{-8}$ |
| 29 | 3 | 1826 | $< 10^{-8}$ | 159 | $< 10^{-8}$ |
| 30 | 3 | 438 | $< 10^{-8}$ | 1199 | $4.10^{-8}$ |
| 31 | 3 | 575 | $8.10^{-8}$ | 576 | $< 10^{-8}$ |
| 32 | 3 | 200 | $< 10^{-8}$ | 874 | $< 10^{-8}$ |
| 33 | 3 | 188* | $2.10^{-8}$ | 672* | $3.10^{-7}$ |
| 36 | 3 | 250 | $< 10^{-8}$ | 263 | $2.10^{-6}$ |
| 45 | 5 | 290 | $< 10^{-8}$ | 369 | $< 10^{-8}$ |
| 52 | 5 | 1357 | $8.10^{-8}$ | 374 | $< 10^{-8}$ |
| 55 | 6 | 316* | $< 10^{-8}$ | 581* | $3.10^{-8}$ |
| 56 | 7 | 1278 | $7.10^{-7}$ | 446 | $< 10^{-8}$ |
| 60 | 3 | 733 | $< 10^{-8}$ | 347 | $1.10^{-8}$ |
| 61 | 3 | 262 | $< 10^{-8}$ | 217 | $< 10^{-8}$ |
| 63 | 3 | 353 | $< 10^{-8}$ | 298 | $< 10^{-8}$ |
| 65 | 3 | 544 | $1.10^{-7}$ | – | fails |
| 71 | 4 | 1173 | $< 10^{-8}$ | 1846 | $5.10^{-3}$ |
| 72 | 4 | 1696 | $1.10^{-8}$ | 1606 | $5.10^{-2}$ |
| 76 | 4 | 1409 | $< 10^{-8}$ | 424 | $< 10^{-8}$ |
| 78 | 5 | 205 | $< 10^{-8}$ | 278 | $< 10^{-8}$ |
| 80 | 5 | 192 | $< 10^{-8}$ | 1032 | $2.10^{-8}$ |
| 81 | 5 | 220 | $< 10^{-8}$ | 1662 | $5.10^{-7}$ |
| 104 | 8 | 1558 | $5.10^{-8}$ | 1109 | $< 10^{-8}$ |
| 108 | 9 | 1106 | $2.10^{-8}$ | 984 | $7.10^{-5}$ |

*converges to a *local*, nonglobal minimum.

and where for each overshoot point a half step backwards is taken, from where the trajectory is continued at reduced speed. Restarting from $\mathbf{x}^*(\mu_0)$, and with $\mu$ increased to $\mu_1$, Phase 1 generates a new trajectory that converges to a much more accurate approximation $\mathbf{x}^*(\mu_1) = [0.636; 0.318]^\top$ with $f(\mathbf{x}^*(\mu_1)) = 0.607$. At this point, since $g(\mathbf{x}^*(\mu_1)) > 0$, the algorithm identifies $g(\mathbf{x}) = 0$ as an active constraint. In the final Phase 2, starting from $\mathbf{x}^*(\mu_1)$, a path is computed that in effect seeks the nearest point on the constraint $g(\mathbf{x}) = 0$. This point $\mathbf{x}^c = [0.659; 0.341]^\top$, with $f(\mathbf{x}^c) = 0.667$, is taken as the final approximation to $\mathbf{x}^*$.

## 8. RESULTS FOR TEST PROBLEMS

As has been stated in the introduction, the LFOPC algorithm has already been thoroughly tested by its implementation on various problems of practical interest [14–17]. It is in these problems, where accurate analytical gradients are not generally available, and where the optimization problems may not always be well defined, that the robustness and reliability of the method comes to the fore. This is particularly so, when it is used as a solver of analytically simple optimization subproblems constructed from the sampling of the behavior of large and complicated systems that have to be optimized. In these cases, the number of iterations required to solve the relatively simple subproblem is of secondary importance since, relative to the costly analyses required to construct the subproblem, the computational cost of evaluating the subproblem functions is negligible.

The purpose of testing the performance of LFOPC on analytical test problems is therefore primarily to demonstrate its reliability and accuracy, rather than its speed of convergence. Table 1 lists the results obtained when applying LFOPC to 36 different problems, arbitrarily selected from the famous set of test problems of Hock and Schittkowski [13]. The dimension $n$ of the problems ranges from 2 to 9, with the total number of constraints per problem varying from 1 to 22. For all the problems considered here, the standard settings given in Section 5(i) and (ii) are used. For the maximum step size the default setting, $\delta = 1$, was mainly used. No normalization of $f(\mathbf{x})$ (see Section 5(iii)), was done for any of the problems. For LFOPC the number of steps, i.e., the number of penalty function gradient evaluations = NPGE, required for convergence is listed together with the relative error, given by $|f(\mathbf{x}^*) - f(\mathbf{x}^c)|/(1 + |f(x^*)|)$, where $\mathbf{x}^c$ is the approximation to $\mathbf{x}^*$ at convergence.

As a matter of interest, results are also listed for one of the best implementations of the classical penalty function method, namely that of the code FMIN due to Kraft and Lootsma in [13]. For their code, the required number of objective function evaluations (NFE) are listed together with the relative error at convergence. The main convergence criterion used by FMIN is that the maximum constraint violation should be less than $10^{-7}$. Although both algorithms are based on penalty function formulations of the constrained problems, they differ drastically from each other, with LFOPC requiring no objective function evaluations and no line searches. For the smooth analytical test functions considered here, there does not seem to be much difference in reliability between the respective methods, although the results seem to marginally favor LFOPC. Should, however, noise be present in the objective and constraint functions, one may expect from the previous evidence of the insensitivity of the LFOP algorithm to noise and inaccuracies of gradients that LFOPC will succeed where other traditional gradient methods, using line searches, break down.

## 9. CONCLUSION

An accurate and reliable algorithm for solving constrained optimization problems, through the application of the well-established leap-frog unconstrained optimization method to penalty function formulations of the original constrained problems, has successfully been developed. The algorithm automatically executes normalization and scaling operations on the gradients of the constraints, resulting in an algorithm requiring virtually no parameter settings.

From the evidence presented here, and from experience already gained separately through the application of the method to practical real-life problems, the indications are that this method is extremely robust being relatively insensitive to inaccuracies in the gradients. Furthermore, in the case where the optimization problem may be inconsistent with no feasible region existing, the method automatically yields a compromised solution, which may be of great value, without resorting to any specialized procedure. These robust characteristics make this method ideally suitable as a standard solver of successive approximate subproblems, when a successive approximation procedure is to be adopted in optimizing large and complicated systems.

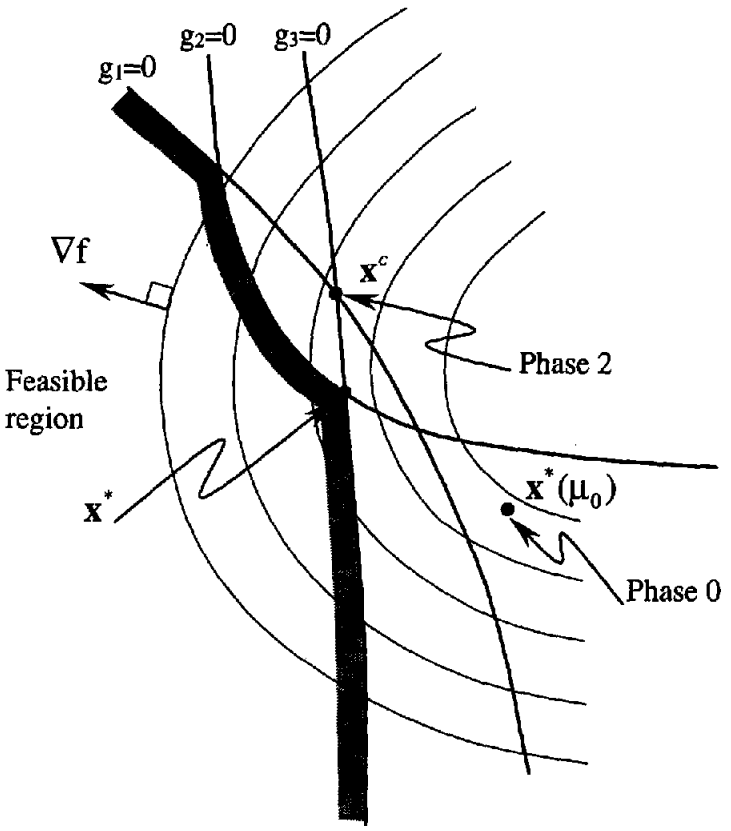



Figure A. Possible failure of LFOPC if Phase 1 is bypassed for the case $n_a = n = 2$.

## APPENDIX

Figure A depicts how, for $n = 2$ and $r = 0$, the wrong set of active inequality constraints may be identified if Phase 1 is bypassed and more than $n$ constraints are almost active at $\mathbf{x}^*$. At convergence to $\mathbf{x}^*(\mu_0)$ in Phase 0, the algorithm identifies active inequality constraints $g_1(\mathbf{x}) = 0$ and $g_3(\mathbf{x}) = 0$. Since the number of active constraints is equal to $2 = n$, Phase 0 is bypassed and Phase 2 solves for the intersection $\mathbf{x}^c$ of the two identified active constraints, which is then taken as the computed approximation to $\mathbf{x}^*$. However, clearly at $\mathbf{x}^c$ a constraint violation occurs with $g_2(\mathbf{x}^c) > 0$. Indeed, the true solution $\mathbf{x}^*$ lies at the intersection of $g_2(\mathbf{x}) = 0$ and $g_3(\mathbf{x}) = 0$. Although the situation depicted here is mathematically possible, its occurrence in practice appears to be very rare if the standard parameter settings are used. If it should occur, it is easily detected from a constraint violation at $\mathbf{x}^c$ where none is expected. In such a case, the problem should be rerun with Phase 1 enforced.

## REFERENCES

1. J.A. Snyman, A new and dynamic method for unconstrained minimization, *Appl. Math. Modelling* **6**, 449–462, (1982).
2. J.A. Snyman, An improved version of the original leap-frog dynamic method for unconstrained minimization LFOP1(b), *Appl. Math. Modelling* **7**, 216–218, (1983).
3. J.A. Snyman, N. Stander and W.J. Roux, A dynamic penalty function method for the solution of structural optimization problems, *Appl. Math. Modelling* **8**, 453–460, (1994).
4. J.E.W. Holm and E.C. Botha, Leap-frog is a robust algorithm for training neural networks, *Network: Computation in Neural Systems* **10**, 1–13, (1999).
5. J.A. Snyman, C. Frangos and Y. Yavin, Penalty function solutions to optimal control problems with general constraints via a dynamic optimisation method, *Computers Math. Applic.* **23** (11), 47–56, (1992).
6. J.A. Snyman, LFOPCON: A general mathematical programming FORTRAN code, Department of Mechanical Engineering, University of Pretoria, Pretoria, South Africa (1993).
7. D.F. Berner and J.A. Snyman, The influence of joint angle constraints on the optimum design of a manipulator following a complicated path, *Computers Math. Applic.* **37** (1), 111–124, (1999).

8. J.A. Snyman and D.F. Berner, A mathematical optimization methodology for the optimal design of a planar robotic manipulator, *International Journal for Numerical Methods in Engineering* **44**, 535–550, (1999).
9. J.A. Snyman and D.F. Berner, The design of a planar robotic manipulator for optimum performance of prescribed tasks, *Structural Optimization* **18**, 95–106, (1999).
10. J.A. Snyman and F. van Tonder, Optimum design of a three dimensional robotic manipulator, *Structural Optimization* **17**, 172–185, (1999).
11. R.J. Minnaar, Optimal dimensional synthesis of planar mechanisms, Master of Engineering thesis, Department of Mechanical Engineering, University of Pretoria, Pretoria, South Africa (1999).
12. R. Fellini, Derivative-free and global search optimization algorithms in an object-oriented design framework, Master of Science thesis, Department of Mechanical Engineering and Applied Mechanics, University of Michigan, Ann Arbor, MI, (1998).
13. W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Springer-Verlag, Berlin, (1981).
14. K.J. Craig, D.J. de Kock and J.A. Snyman, Using CFD and mathematical optimisation to minimize stack pollution, *International Journal for Numerical Methods in Engineering* **44**, 551–566, (1999).
15. K.J. Craig, P. Venter, D.J. de Kock and J.A. Snyman, Optimization of structured grid spacing parameters for separated flow simulation using mathematical optimization, *Journal of Wind Engineering and Industrial Aerodynamics* **80**, 221–231, (1999).
16. D.J. de Kock, K.J. Craig and J.A. Snyman, Using mathematical optimisation in the CFD analysis of a continuous quenching process, *International Journal for Numerical Methods in Engineering* **47**, 985–999, (2000).
17. L.J. du Plessis, An optimization approach to the determination of manipulator workspaces, Master of Engineering thesis, Department of Mechanical Engineering, University of Pretoria, Pretoria, South Africa, (1999).
18. P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization*, Academic Press, New York, (1981).